tls

Why TLS?

- TLS (Transport Layer Security) offers a secure connection between a client and server.
- What happens if you want to login to a website with a username and password, or enter your credit card details?
- Without encryption, this can be taken by an attacker who has hijacked a connection between the client and server.

Attacks on HTTP

- An attacker who listens to your network connection (e.g. a public one) can read all transmissions in plain text and recover them.
- They can also impersonate the server, and take your data that way.

The Rules of the Game

- Assume that any attacker owns the connection between a client and a server.
- They can thus intercept and impersonate any messages on said connection.
- Given this, we want to have a *secure* connection between client and server.

A Simple Scheme

- Use RSA encryption: The client and server both generate a public and private key and exchange the public key over the network.
- They then send messages by encrypting each other's public key to encrypt the payload and then only those with the private key can decrypt the message.
- This works, right?

A Simple Attack

- The attacker has to do the following:
- First, prepare two pairs of RSA keys. We'll call these Pub_1, Pri_1, Pub_2, Pri_2.
- Let's call the Client's keys Pub_C and Pri_C.
- Let's call the Server's keys Pub_S and Pri_S.
- When the client sends their public key to the server, intercept and keep it.
  - Send the server Pub_1.
- When the server sends their public key to the client, intercept and keep it.
  - Send the client Pub_2.

A Simple Attack Cont.

- When the client sends a message, decrypt it using Pri_1, since they encrypted it with Pub_1. Send it over untampered.
- When the server sends a message, decrypt it using Pri_2, since they encrypted it with Pub_2. Send it over untampered.
- You can now listen to any message between this connection forever.
- You can also use the client + server's public keys to impersonate them, since they are used to encrypt messages from them.

Diffie-Hellman

- RSA fails for this use-case. Lets use another method.
- The client chooses a value a where $A = g^a \mod p$. They send A.
- The server chooses a value b where $B = g^b \mod p$. They send B.
- They both compute $K = B^a \% p = g^{ab} \mod p$.
- They both compute $K = A^b \% p = g^{ab} \mod p$.
- K (the secret value) cannot be found out easily with just A and B.

Impersonation

- How does a client **know** they're talking to the right server?
- If someone can impersonate the server, they can steal information from any client by posing as a website.
- There needs to be some way of attesting that a website is who they say they are.
- We can still do the previous attack on a Diffie-Hellman Key Exchange, even though the secret is not leaked – there's no need to know the secret, just do a separate key exchange with the client and server and send their data off to each other.

Certificate Authorities (CAs)

- We need some way to attest a website is who they say they are. We need to certify websites.
- We can have 3rd parties attest that a website is who they say they are, and then give out binding agreements (Certs). We then use these to provide identity checks.

TLS

- With the certificate check, a client can know that it's talking to the right website.
- It can then do a Diffie-Hellman key exchange even on a tapped connection.
- And any subsequent messages can be encrypted using a secret transferred by DH key exchange, allowing for encryption throughout the connection.

Questions?